(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－232079

(43)公開日　平成11年(1999)８月27日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　7/00		Ｇ０６Ｆ　7/00	Ｅ
17/50		15/60	６５４Ｚ

審査請求　未請求　請求項の数２　ＯＬ　（全 12 頁）

(21)出願番号　特願平10－32673

(22)出願日　平成10年(1998)２月16日

(71)出願人　000005496
富士ゼロックス株式会社
東京都港区赤坂二丁目17番22号

(72)発明者　山田　紀一
神奈川県足柄上郡中井町境430　グリーンテクなかい富士ゼロックス株式会社内

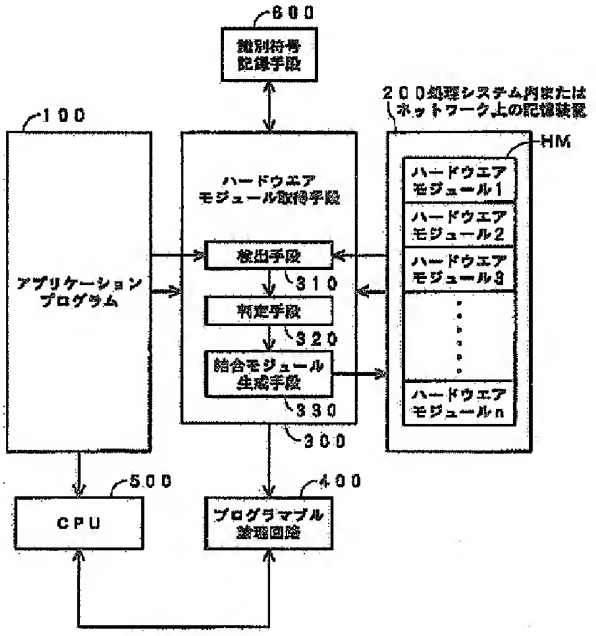
(74)代理人　弁理士　佐藤　正美

(54)【発明の名称】　情報処理システム

(57)【要約】

【課題】　少なくとも処理の一部分が、プログラマブル論理回路で処理される情報処理システムにおいて、アプリケーションプログラムを高速に処理することができるプログラマブル論理回路の構成方法を提供する。

【解決手段】　処理を前記プログラマブル論理回路に再構成する回路情報で記述したハードウエアモジュールを、情報処理システム内記憶装置、または、ネットワーク上の記憶装置から取得して、プログラマブル論理回路に再構成するものである。プログラム中で処理順序が連続する複数のハードウェアモジュールの組み合わせを検出し、その組み合わせよる結合回路が、プログラマブル論理回路に同時に構成可能かを判定する。その判定の結果、プログラマブル論理回路に同時に構成可能な、処理順序が連続する複数のハードウェアモジュールを結合して１つの新ハードウェアモジュールを生成し、情報システム内記憶装置に格納し、それによりプログラマブル論理回路をコンフィギュレーションできるようにする。



1

【特許請求の範囲】

【請求項１】アプリケーションプログラムにより実行される複数の処理の少なくとも一部分の処理を、プログラマブル論理回路で処理するものであって、前記処理を前記プログラマブル論理回路に再構成する回路情報で記述したハードウエアモジュールを、情報処理システム内記憶装置、または、ネットワーク上の記憶装置から取得して、前記プログラマブル論理回路に再構成するようにする情報処理システムにおいて、

前記アプリケーションプログラム中で処理順序が連続する複数のハードウェアモジュールの組み合わせを検出する検出手段と、

前記検出手段で検出された処理順序が連続する複数のハードウェアモジュールの組み合わせによる結合回路が、前記プログラマブル論理回路に同時に構成可能かを判定する判定手段と、

前記判定手段で前記プログラマブル論理回路に同時に構成可能な、前記処理順序が連続する複数のハードウェアモジュールを結合して１つの新ハードウェアモジュールを生成し、前記情報システム内記憶装置に格納する結合ハードウエアモジュール生成手段と、

を備え、

前記アプリケーションプログラムの実行時に、実行しようとする処理の後続の処理の組み合わせに応じて、前記結合ハードウエアモジュールにより、前記プログラマブル論理回路を再構成することを特徴とする情報処理システム。

【請求項２】請求項１に記載の情報処理システムにおいて、

前記検出手段は、前記アプリケーションプログラム中で処理順序が連続する複数のハードウェアモジュールの組み合わせの前記プログラム中での出現頻度をも検出するものであり、

前記判定手段は、前記検出手段で検出された処理順序が連続する複数のハードウェアモジュールの組み合わせのうち、前記検出された出現頻度の高いものについて、その組み合わせによる結合回路が、前記プログラマブル論理回路に同時に構成可能かを判定することを特徴とする情報処理システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】この発明は、少なくとも処理の一部分を、回路構成を再構成できるプログラマブル論理回路で処理することが可能である情報処理システムに関するものである。

【０００２】

【従来の技術】デジタル回路装置、特に特定用途向け集積回路（ＡＳＩＣ）の分野において、製品の開発期間を短縮するために、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）やプログラマブルロジックデバイス（ＰＬＤ）などで構成されたプログラマブル論理回路が広く使われている。

2

【０００３】これらのプログラマブル論理回路は、論理回路を記述する回路情報をこれらに読み込ませることで、内部の論理回路と論理回路間の結線を自由に構成することができる。このため、プログラマブル論理回路装置を用いることで、従来は回路設計の終了後に数週間から数か月を必要とした集積回路の作製時間が不要になるというメリットがある。特に、米国特許第４，７００，１８７号の発明のような電気的に再構成可能なプログラマブル論理回路装置は、一度作製した回路を必要に応じて自由に何度でも変更できるという利点があり、プログラマブル論理回路装置は、ますます広く使われるようになってきている。

【０００４】ところで、最近の論理回路は複雑さが増し、一つのプログラマブル論理回路装置では実現できない規模にまで回路規模が大きくなっている。

【０００５】この問題を解決するためのひとつの方法として、異なる時間に異なる論理回路を実現するためにプログラマブル論理回路を処理の途中で再構成することが提案されている。この方法を用いることにより、携帯情報端末のように、装置が小型であるため、内蔵できる回路規模に制約がある場合でも、様々な処理が比較的高速に行えるという利点がある。

【０００６】しかし、プログラマブル論理回路を再構成するときには、回路全体の回路情報を再度読み込ませるため、再構成に時間がかかるという欠点がある。さらに、処理の途中で再構成することは、処理を一時中断し、その時のデータをプログラマブル論理回路の外部の記憶装置に待避させ、新たな回路情報を読み込んで再構成し、再構成前のデータと再構成に伴う新しいデータを入力するという余分な処理が必要で、データを出し入れする処理は冗長なものとになる。

【０００７】この問題を解決するために、米国アトメル社の「ＣＯＮＦＩＧＵＲＡＢＬＥＬＯＧＩＣ」という名のデータブックに記載されているプログラマブル論理回路、および米国ザイリンクス社の「ＴＨＥ　ＰＲＯＧＲＡＭＭＡＢＬＥ　ＬＯＧＩＣ」という名のデータブックに記載されているプログラマブル論理回路では、データを記憶するためのデータ記憶装置を有し、回路の動作中でも外部の記憶装置から回路情報の一部を読み込んで部分的に再構成を行うことで、再構成するための時間を最小に留めるようにしている。

【０００８】以上のようなプログラマブル論理回路を用いて、高速かつ簡便に処理することができる計算機システムが、特開平６－３０１５２２号公報に開示されている。この計算機システムは、複数のプログラムを元に、プログラマブル論理回路上にハードウエアを構成する例である。これを、図１７を参照しながら説明する。

【０００９】すなわち、図１７の従来例においては、計

算機で実行するソースプログラム１０００は、回路構成を変更できない固定部と、プログラマブル論理回路のように回路構成を変更できる可変部とで構成される。ライブラリ１００１には、固定部の構成に関する情報と、可変部が構成することができる回路の情報が格納されている。

【００１０】コンパイラ１００２は、ソースプログラム１０００を解析し、ライブラリ１００１を参照しながら、オブジェクトコードと、ハードウエア構成データに変換する。例えば、コンパイラ１００２は、ソースプログラムのフロー解析を行い、関数の頻度を検出し、その検出した頻度に基づいて、呼び出し回数の多い関数をハードウエアで処理する関数として決定し、ハードウエア構成データ１００３を作成し、出力する。

【００１１】次に、コンパイラ１００２は、ハードウエアで処理すると決めた部分を所定の可変部で処理することを示すコードを生成する。そして、このコードを、残りのソフトウエアで処理する部分に付加してオブジェクトコード１００４を作成し、出力する。計算機１００５は、固定部と、ハードウエア構成データにより構成された可変部とを用いて、オブジェクトコードに応じた処理を実行する。

【００１２】このようにして、従来例１では、コンパイル時に呼び出し回数の多い関数をハードウエア化することにより処理全体の高速化を図っている。

【００１３】

【発明が解決しようとする課題】上述した従来例のように、一般に、プログラマブル論理回路で実行する処理は、ひとつのプログラム内で、一つの処理と定められ、当該処理は、呼び出し回数の多い関数というように一元的に決められるものであって、プログラマブル論理回路で実行する処理による例えば関数演算が、ひとつのプログラム内で複数になる場合は、その関数の呼び出しごとに回路情報の再構成が必要である。

【００１４】このようにプログラマブル論理回路で実行する処理を、関数の単位で決めて、プログラマブル論理回路に再構成する場合には、次のような問題がある。

【００１５】すなわち、プログラマブル論理回路に再構成する回路情報のサイズは、関数の処理内容によって、小さいものから大きいものまで存在するが、呼び出す関数が頻繁に変わる場合に、それが小さいサイズの回路の連続でも、必ず、呼び出しのたびに、プログラマブル論理回路を再構成する期間が生じ、プログラム全体の処理速度の低下を招くという問題がある。

【００１６】一方、動的に書換え可能なプログラマブル論理回路を利用して、すでに回路が構成されているプログラマブル論理回路の空いている領域に、別の回路情報を構成することもできる。しかし、この場合には、常にプログラマブル論理回路上の空き状況を調べてから追加再構成するか、全面再構成するかを決める処理が介在することになり、やはりプログラム全体の処理速度の低下を招く問題がある。

【００１７】この発明は、上述した従来技術の問題に鑑み、少なくとも処理の一部分が、プログラマブル論理回路で処理される情報処理システムにおいて、アプリケーションプログラムを高速に処理することができるプログラマブル論理回路の構成方法を提供することを目的とするものである。

【００１８】

【課題を解決するための手段】上記課題を解決するため、この発明による情報処理システムは、アプリケーションプログラムにより実行される複数の処理の少なくとも一部分の処理を、プログラマブル論理回路で処理するものであって、前記処理を前記プログラマブル論理回路に再構成する回路情報で記述したハードウエアモジュールを、情報処理システム内記憶装置、または、ネットワーク上の記憶装置から取得して、前記プログラマブル論理回路に再構成するようにする情報処理システムにおいて、前記アプリケーションプログラム中で処理順序が連続する複数のハードウェアモジュールの組み合わせを検出する検出手段と、前記検出手段で検出された処理順序が連続する複数のハードウェアモジュールの組み合わせによる結合回路が、前記プログラマブル論理回路に同時に構成可能かを判定する判定手段と、前記判定手段で前記プログラマブル論理回路に同時に構成可能な、前記処理順序が連続する複数のハードウェアモジュールを結合して１つの新ハードウェアモジュールを生成し、前記情報システム内記憶装置に格納する結合ハードウエアモジュール生成手段と、を備え、前記アプリケーションプログラムの実行時に、実行しようとする処理の後続の処理の組み合わせに応じて、前記結合ハードウエアモジュールにより、前記プログラマブル論理回路を再構成することを特徴とする。

【００１９】また、請求項２の発明は、請求項１に記載の情報処理システムにおいて、前記検出手段は、前記アプリケーションプログラム中で処理順序が連続する複数のハードウェアモジュールの組み合わせの前記プログラム中での出現頻度をも検出するものであり、前記判定手段は、前記検出手段で検出された処理順序が連続する複数のハードウェアモジュールの組み合わせのうち、前記検出された出現頻度の高いものについて、その組み合わせによる結合回路が、前記プログラマブル論理回路に同時に構成可能かを判定することを特徴とする。

【００２０】

【作用】上述の構成の発明による情報処理システムにおいては、アプリケーションプログラム中で処理順序が連続する複数のハードウェアモジュールの組み合わせであって、その結合が領域的にプログラマブル論理回路に再構成可能な場合には、その複数のハードウェアモジュールが結合されて１つの新ハードウェアモジュールが予め

生成され、システム内記憶装置に格納される。

【００２１】そして、アプリケーションプログラムの実行時には、連続する複数の処理の実行開始前に、その連続する複数の処理に対応する新ハードウエアモジュールがシステム内記憶装置から読み出されて、プログラマブル論理回路に再構成される。

【００２２】したがって、複数の処理であっても、結合された新ハードウエアモジュールを用いる場合には、その複数の処理についてのプログラマブル論理回路の再構成は１回で良くなり、これにより、アプリケーションプログラムにより実行される全体の処理の中での再構成回数が少なくなり、処理時間の短縮化が図れる。

【００２３】請求項２の発明の場合には、特に、連続する複数の処理の組み合わせの出現頻度が多いもの、つまり、繰り返し使用される組み合わせについて、新ハードウエアモジュールが生成されて、システム内記憶装置に格納されて、アプリケーションプログラムの実行時に利用されるので、再構成回数の削減効果が大きくなり、プログラマブル論理回路を効率良く使用した処理ができる。

【００２４】

【発明の実施の形態】以下、この発明による情報処理システムの実施の形態を、図を参照しながら説明する。

【００２５】この発明による、少なくとも処理の一部分が回路構成を再構成できるプログラマブル論理回路で処理される情報処理システムの実施の形態の主要な概念構成を図１に示す。

【００２６】この実施の形態において、対象とするアプリケーションプログラム（以下の説明においては、単にプログラムという場合もある）１００は、当該プログラムにより実行しようとする一連の処理を複数個の処理に分離し、分離された各処理ごとにモジュールとして構成したものである。このモジュールを、この明細書では処理モジュールと称することとする。

【００２７】この実施の形態においては、アプリケーションプログラム１００は、その一部または全部の処理モジュールを、プログラマブル論理回路４００によるハードウエア処理により行うように記述するものである。プログラマブル論理回路４００で処理を行わない処理モジュールは、ＣＰＵ５００によるソフトウエア処理とされる。

【００２８】プログラマブル論理回路に再構成する回路情報で記述したモジュール（この明細書では、このモジュールをハードウエアモジュールと称する）ＨＭは、ネットワーク上の記憶装置２００あるいは当該情報処理システム内の記憶装置に保存して用意する。

【００２９】この例の場合、プログラム１００は、図２に示すように、ヘッダ部ＨＥＤと、本体部ＰＲＧとからなり、本体部ＰＲＧには、プログラムが記述されている。そして、ヘッダ部ＨＥＤには、使用するハードウエアモジュールＨＭの識別符号が、その使用順に記述されている。また、各ハードウエアモジュールＨＭの識別符号に対応して、各ハードウエアモジュールＨＭのファイルサイズが記述されている。

【００３０】このプログラム１００に記述されているハードウエアモジュールＨＭの識別符号とファイルサイズの情報は、識別符号記録手段６００に記録される。

【００３１】ハードウエアモジュール取得手段３００は、プログラム１００中に記述されている識別符号ＩＤを用いて、ハードウエアモジュールＨＭを、情報処理システム内またはネットワーク上の記憶装置２００から取得する。そして、ハードウエアモジュール取得手段３００は、取得したハードウエアモジュールＨＭによる回路をプログラマブル論理回路４００上に再構成する。

【００３２】そして、この実施の形態の場合、ハードウエアモジュール取得手段３００は、アプリケーションプログラム１００中で処理順序が連続する複数のハードウェアモジュールの組み合わせを検出する検出手段３１０と、この検出手段３１０で検出された処理順序が連続する複数のハードウェアモジュールの組み合わせによる結合回路が、プログラマブル論理回路４００に同時に構成可能かを判定する判定手段３２０と、この判定手段３２０でプログラマブル論理回路４００に同時に構成可能な、処理順序が連続する複数のハードウェアモジュールを結合して１つの新ハードウェアモジュールを生成し、情報システム内記憶装置２００に格納する結合モジュール生成手段３３０とを備える。

【００３３】これら検出手段３１０、判定手段３２０および結合モジュール生成手段３３０は、ハードウエアモジュール取得手段３００において、プログラム実行前の前処理として機能する。

【００３４】検出手段３１０は、この例の場合には、プログラム１００のヘッダ部ＨＥＤの識別符号ＩＤの並び順により、処理順序が連続する複数のハードウェアモジュールの組み合わせを検出する。

【００３５】判定手段３２０は、検出手段３１０で検出された連続する複数のハードウェアモジュールの組み合わせによる結合回路が、プログラマブル論理回路４００に同時に構成可能かを、ヘッダ部ＨＥＤのファイルサイズと、プログラマブル論理回路４００のコンフィギュレーションメモリ領域とから判定する。

【００３６】結合モジュール生成手段３３０は、判定手段３２０でプログラマブル論理回路に再構成可能なサイズの結合モジュールのみを生成する。そして、結合モジュール生成手段３３０は、生成した結合モジュールについて、新しい識別符号を付与し、その識別符号を識別符号記録手段６００に記録する。

【００３７】そして、ハードウエアモジュール取得手段３００は、この前処理において生成された結合モジュールをも用いて、識別符号記録手段に記録されている識別

符号に対応するハードウエアモジュールを、システム内記憶装置またはネットワーク上の記憶装置２００から取得して、プログラマブル論理回路４００の再構成を実行する。そして、そのプログラマブル論理回路４００に構成された回路を用いられて、処理が実行される。

【００３８】［実施の形態のハードウエア構成例］図３は、この発明の第１の実施の形態の情報処理システム１０のハードウエア構成例を示すブロック図である。この実施の形態の情報処理システム１０においては、ＣＰＵ１１のホストバス１１Ｂに、チップセット１２に含まれるメモリコントローラ（図示せず）を介して、例えばＤＲＡＭで構成されるメインメモリ１３が接続される。

【００３９】ホストバス１１Ｂは、また、チップセット１２に含まれるホスト－ＰＣＩバスブリッジ（図示せず）を介して、ＰＣＩバス１４に接続される。ＰＣＩバス１４には、プログラマブル論理回路インターフェース１５を介してプログラマブル論理回路１６と、ハードディスクインターフェース１７を介してハードディスクドライブ１８と、通信インターフェース１９とが接続される。

【００４０】通信インターフェース１９は、ＬＡＮやインターネットなどのネットワーク２０を介して、プログラマブル論理回路１６に再構成される回路情報が格納されている記憶装置（サーバ）２１に接続される。

【００４１】ハードディスクドライブ１８により読み書きされるハードディスクには、アプリケーションプログラムが格納されている。また、アプリケーションプログラムは、ネットワーク２０上の記憶装置に格納されている場合もある。

【００４２】また、ハードディスクドライブ１８のハードディスクやメインメモリ１３には、ネットワーク２０上の記憶装置２１から取得されたハードウエアモジュールＨＭが格納される。また、前述したように生成された結合モジュールも、ハードディスクドライブ１８のハードディスクやメインメモリ１３に記憶される。すなわち、ハードディスクドライブ１８やメインメモリ１３は、システム内記憶装置を構成する。

【００４３】この実施の形態では、ハードウエアモジュール取得手段３００が、図３で示した情報処理システム１０のＯＳのひとつの機能としてソフトウエア的に実装される。また、識別符号記録手段６００は、ハードディスクドライブ１８やメインメモリ１３で構成される。

【００４４】次に、プログラマブル論理回路１６の構造を図４に示す。プログラマブル論理回路１６は、図５に示すように、回路情報を格納するためのコンフィギュレーションメモリ１６０と、論理セル１６１と、配線領域１６２と、入出力端子１６３とで構成される。

【００４５】コンフィギュレーションメモリ１６０は、論理セル１６１内および配線領域１６２内のＳＲＡＭ、ＤＲＡＭなどの書き換え可能なメモリ素子で構成されている。コンフィギュレーションメモリ１６０にアドレスＡＤＲが与えられて、新しい回路情報のデータＤＡが格納されると、この回路情報に従って、論理セル１６１内の回路構成と、論理セル１６１および入出力端子１６３を相互に接続する配線領域１６２の接続状態が再構成される。この一連の動作をコンフィギュレーションと呼ぶ。コンフィギュレーションメモリ１６０の一部分を書き換えることで、プログラマブル論理回路が動作中であっても、回路を部分的に再構成することができる。

【００４６】図５に示すように、プログラマブル論理回路１６に再構成されて形成された回路素子１６４に、処理すべきデータが入力され、また、その処理結果が出力される。

【００４７】［この発明の実施の形態による処理の説明］図６は、この発明の実施の形態における基本的な処理の流れを示すフローチャートである。

【００４８】実際のアプリケーションプログラムの実行に先立ち、ハードウエアモジュール取得手段３００は、コンフィギュレーションデータ前処理のルーチンＲ１００を実行する。

【００４９】まず、図２に示したようにプログラム１００内のヘッダ部ＨＥＤに記述されているハードウエアモジュールＨＭの識別符号ＩＤと、そのハードウェアモジュールの回路規模を表すファイルサイズデータが読み込まれ、当該アプリケーションプログラムで使用される、プログラマブル論理回路１６上に構成する必要のあるハードウエアモジュールＨＭのすべてが、ハードウエアモジュール取得手段３００で認識される（ステップＳ１０１）。

【００５０】次に、構成すべき複数ハードウェアモジュールＨＭの２つ以上の組合せを全て算出し、それぞれの組合せのハードウェアモジュールのサイズデータの合計と、プログラマブル論理回路に搭載し得る最大回路規模とを比較して、その組み合わせのハードウエアモジュールがプログラマブル論理回路に搭載可能かどうかを判断する（ステップＳ１０２）。

【００５１】図７は、処理順序が連続する結合可能なハードウエアモジュールの組み合わせ、およびその結合後のサイズがプログラマブル論理回路に搭載可能かどうかの情報として、ステップＳ１０２で算出された結合可能テーブルＴＢＬ１の例を示すものである。

【００５２】図７において、アプリケーションプログラムで必要とするハードウェアモジュールＨＭの数がＮ個とすると、連続する組合せは、２個のハードウエアモジュールの組み合わせからＮ個のハードウエアモジュールの組み合わせまで考えられる。なお、処理の目的に応じてハードウェアモジュールの実行順序が定型化している場合は、検索するこれらの組み合わせの種類を限定してもよい。

【００５３】それぞれの連続する組み合わせのハードウ

エアモジュール個数（図７の連続数）に対し、ハードウェアモジュールの識別符号（図の例では、Ａ，Ｂ，Ｃ，Ｄ，…）を用いて組み合わせ名を表現している。そして、あらかじめプログラム１００から取得したハードウェアモジュールＨＭのデータサイズから、結合した場合のデータサイズを算出し、その結合後のデータサイズと、プログラマブル論理回路上に搭載できる最大サイズとを比較して、その比較結果を、プログラマブル論理回路に搭載できるか否かの情報として結合可能テーブルＴＢＬ１に記載している。

【００５４】この図７の結合可能テーブルＴＢＬ１においては、プログラマブル論理回路に搭載できるか否かの情報としては、結合後のデータサイズが、プログラマブル論理回路上に搭載できる最大サイズより小さいならば、その組み合わせのハードウェアモジュールは同時にプログラマブル論理回路上に搭載可能と判断して、図７では「○」印で示すような判断情報を記録する。また、結合後のデータサイズが、プログラマブル論理回路上に搭載できる最大サイズより大きいならば、その組み合わせのハードウェアモジュールは同時にプログラマブル論理回路上に搭載不可能と判断して、図７では「×」印で示すような判断情報を記録する。

【００５５】図７では、ハードウェアモジュールの識別符号をＡ，Ｂ，Ｃ，Ｄ，Ｅの５種とし、ＡＢ，ＡＣ，ＡＢＣの組み合わせは同時に搭載可能だが、ＡＤ，ＡＥ，ＡＢＥといった組み合わせは同時に搭載できないと判断されている。

【００５６】以上のような、結合可能な組み合わせ算出処理が終了すると、次に、実行処理を行うアプリケーションプログラム中で、プログラマブル論理回路によって処理を行う部分の処理順序を解析し、ループの回数なども含めて連続するハードウェアモジュール処理順序の組み合わせを抽出して、その組み合わせの出現頻度を算出する（ステップＳ１０３）。

【００５７】この算出結果の頻度テーブルＴＢＬ２の例を図８に示す。この図８の頻度テーブルＴＢＬ２は、プログラム中から抽出された組み合わせを、頻度の高い順に並び替えたものである。この図８の例では、例えば、ＡＢという順序の組み合わせ処理がアプリケーションプログラム中で１００回出現し、ＡＣという順序の組み合わせ処理がアプリケーションプログラム中で１００回出現していることを示している。

【００５８】次に、出現頻度の高い処理順序の組み合わせから順に、先に算出したプログラマブル論理回路上に同時に搭載可能なハードウェアモジュールの組み合わせと照合し、その組み合わせのハードウエアモジュールが搭載可能なら、そのハードウェアモジュールの回路構成データを結合し（ステップＳ１０４）、ひとつのコンフィギュレーションデータとして情報処理システム内の記憶装置に記憶し、新規な識別符号を与えてその識別符号は識別符号記録手段６００に記録する（ステップＳ１０５）。以上で、前処理のルーチンＲ１００が終了する。

【００５９】図９は、ステップＳ１０４のハードウエアモジュールの結合処理ルーチンの詳細例を示すフローチャートである。

【００６０】すなわち、このルーチンにおいては、まず、一連の繰り返し処理数をカウントする変数ｉが初期化される（ステップＳ３０１）。次に、プログラムから抽出した処理順序が連続するハードウエアモジュールの組み合わせごとの出現頻度の情報として、ステップＳ１０３で算出した図８に示したような頻度テーブルＴＢＬ２に基づき、そのｉ番目の頻度の組み合わせは結合可能かどうかについて、ステップＳ１０２で算出した結合可能テーブルＴＢＬ１と逐次照合を行う（ステップＳ３０２）。

【００６１】結合可能と判定された場合は、実際の結合処理を行う（ステップＳ３０３）。通常、プログラマブル論理回路のコンフィギュレーションデータは、チップ内部の論理セルの配置に対応して、アドレスとデータという組み合わせの一連のストリームという形式であり、ハードウェアモジュールの結合処理は、アドレスの相対化を行い、それぞれのモジュールで使用する論理セルが重複しないよう配置変更を行うことで施される。

【００６２】結合したハードウェアモジュールには新規に識別符号を与え、その結合ハードウエアモジュールのデータは、システム内記憶装置に記憶し、その識別符号は識別符号記録手段６００に記録されて、ハードウェアモジュール取得手段３００によって、その管理下に置かれる。

【００６３】ｉ番目の頻度の組み合わせが、結合不可能な時は、結合処理を行わず、それぞれ単独のハードウェアモジュールとして利用することとする。

【００６４】次に、繰り返しの変数ｉをカウントアップし（ステップＳ３０４）、ｉ＋１番目の頻度の組み合わせの結合の処理を、上述したｉ番目の処理と同様行い、ｎ番目の頻度の組み合わせまで処理を繰り返す。

【００６５】以上のようにして、結合可能なハードウェアモジュールは結合され、新しいハードウェアモジュールのセットを使ってのアプリケーションプログラムの実行が開始される。

【００６６】アプリケーションプログラムの実行が開始され、プログラマブル論理回路の処理が呼ばれると、ハードウエアモジュール３００は、コンフィギュレーション実行ルーチンＲ２００の処理を開始する。

【００６７】すなわち、コンフィギュレーション実行ルーチン２００においては、アプリケーションプログラムにより、プログラマブル論理回路で実行すべき処理モジュールが呼ばれるごとに、ハードウエアモジュール取得手段３００は、プログラマブル論理回路上に、その処理モジュールを実行するためのハードウエアモジュールに

よる回路が構成されているかを確認した上で、必要なハードウエアモジュール（結合モジュールを含む）を取得し、プログラマブル論理回路上に再構成する（ステップＳ２０１）。

【００６８】この場合に、アプリケーションプログラムの実行に先立ち、前述したように、実行処理を行うアプリケーションプログラム中で、プログラマブル論理回路によって処理を行う部分の処理順序が解析されているので、プログラマブル論理回路で実行すべき処理モジュールが呼ばれたときには、前記の予め解析されている、その後の処理順序と、新規に生成された結合モジュールの情報とをも参照して、その時に取得するハードウエアモジュールを決定するようにする。この決定は、識別符号により行う。

【００６９】なお、使用できる結合モジュールがないときには、単独のハードウエアモジュールを用いてコンフィギュレーションを行って処理を実行するのはいうまでもない。

【００７０】そして、コンフィギュレーションのステップＳ２０１においては、ハードウエアモジュール取得手段３００は、識別符号記録手段６００に記録されている、先に、プログラム１００から読み込まれた識別符号、また、ハードウェアモジュールの結合によって新規に生成した識別符号のうちの、前記決定された識別符号に対応するハードウエアモジュールの回路情報を、システム内の記憶装置またはネットワーク上の記憶装置から取得する。そして、その取得したハードウエアモジュールの回路情報を、プログラマブル論理回路のコンフィギュレーションメモリに転送し、コンフィギュレーションを実行して、コンフィギュレーションの完了をＯＳ（ＣＰＵ）に通知する。

【００７１】コンフィギュレーションの完了通知があると、プログラマブル論理回路によるハードウエア処理のシーケンスを実行する（ステップＳ２０２）。そして、以上の処理をアプリケーション終了まで繰り返す（ステップＳ２０３）。

【００７２】なお、結合モジュールが選択されたときには、プログラマブル論理回路で実行すべき処理モジュールが呼ばれたときには、既に結合モジュールとしてプログラマブル論理回路に構成されている場合もある。その場合には、ハードウエアモジュール取得手段３００は、ハードウエアモジュールの取得およびコンフィギュレーションを実行することなく、コンフィギュレーションの完了をＯＳに通知する。

【００７３】［実施の形態における処理の具体例］図１０は、この発明の実施の形態におけるコンフィギュレーションと、ハードウェア実行シーケンスを示す模式図である。図１０（Ｂ）が、この発明の実施の形態の実行シーケンスを示すものである。図１０（Ａ）は、結合したハードウエアモジュールを使用せずに、プログラム１００に記述された個々のハードウエアモジュールのみで処理を実行する場合の実行シーケンスを示すもので、これは、この発明の実施の形態の処理との比較例である。

【００７４】図１０の処理の例では、ハードウェアモジュールの処理順序を、Ａ→Ｂ→Ａ→Ｂ→Ｃ→Ｄ→Ａ→Ｃ→Ａ→Ｃ→Ａ→Ｃとしている。

【００７５】図１０（Ａ）の比較例では、まず、ハードウェアモジュールＡをプログラマブル論理回路にコンフィギュレーションし、このハードウエアモジュールＡを用いて処理を実行する。

【００７６】次に、アプリケーションプログラムは、処理Ｂのハードウェアモジュールを要求し、現状のプログラマブル論理回路の構成状態で処理Ｂのハードウェアモジュールが追加搭載可能かどうか、空き状況をチェックする。ここでは、追加搭載構成可能と判断されるので、処理Ｂのハードウェアモジュールをコンフィギュレーションした後、再構成された回路を用いて処理を実行する。

【００７７】続く処理Ａおよび処理Ｂのハードウエアモジュールは、既にプログラマブル論理回路に構成されているので、コンフィギュレーションを行うことなく処理の実行ができる。

【００７８】以下、同様にして、ハードウェアモジュール処理要求が発生するたびに、プログラマブル論理回路上に存在するハードウェアモジュール識別番号のチェックおよび空き状況のチェックと、追加再構成か全面再構成かを判断しながら処理が継続する。

【００７９】これに対して、この発明の実施の形態では、図１０（Ｂ）に示すように、Ａ→Ｂ，Ａ→Ｂ→Ｃ，Ａ→Ｃという処理が連続しており、かつ、同時に、それらの結合ハードウエアモジュールは、プログラマブル論理回路上に搭載できることがわかるので、結合したコンフィギュレーションデータを用いて連続する処理の最初に１回だけコンフィギュレーションを行う。

【００８０】また、プログラマブル論理回路上の空き状況のチェックは、この発明の実施の形態では、ハードウェアモジュールを結合する処理の段階で調べているため、コンフィギュレーションの直前に行う必要がない。コンフィギュレーションデータは、各々の処理の時点で、常に、プログラマブル論理回路上の空き状況を最小にするよう結合してある。

【００８１】アプリケーションプログラムからハードウェアモジュールを使って処理する際は、対応するハードウェアモジュール部分のみ動作させて実行する。その他のハードウェアモジュールはアイドル状態となっている。

【００８２】［処理の実施例］次に、この発明の実施の形態を画像処理に適用した実施例を説明する。

【００８３】画像処理のひとつである空間フィルタ処理は、単位画素から構成される画像ファイルの単一もしく

は複数画素に対して演算を行う。これは、ある画素（データ$x_{l,m}$）の近傍の画素（データ$p_{l,m}$）に、マスクデータ$filter_{l,m}$を掛けて足し合わせ、係数Ｎを乗じたものを処理後の画素値$x_{l,m}$として得る処理として、次の式（１）のように表すことができる。ここで、マスクデータ$filter_{l,m}$や係数Ｎの値を変えることにより、空間フィルタは様々な処理を実行することが可能である。

【００８４】

【数１】

$$x_{l,m}=N\sum_{l-1}^{l+1}\sum_{m-1}^{m+1}p_{l,m}\times filter_{l,m}$$

・・・・式（１）

【００８５】以下に説明する実施例では、２種の画像フィルタを用いた画像処理を行う例である。まず、それら画像フィルタについて説明する。

【００８６】図１１は、２次微分によりエッジを検出するＬａｐｌａｃｉａｎフィルタと呼ばれる画像フィルタに原画像を加えたマスクデータである。このフィルタ処理を施すことにより、画像の輪郭強調を行うことができる。

【００８７】図１２は、図１１のＬａｐｌａｃｉａｎフィルタ処理を実現するソフトウェアを記述したルーチンＳｍＸＹＺ００１のプログラム例を示すものである。また、図１３は、図１１のＬａｐｌａｃｉａｎフィルタ処理を実現するハードウエアモジュールＨｍＸＹＺ００１の回路例を示すものである。

【００８８】図１２は、Ｌａｐｌａｃｉａｎフィルタ処理によって画像の輪郭強調を行うＣプログラムの一例である。演算を施す元の画像データｐ[l][m]に対し、for文を用いた二重ループ構成によって、あらかじめ図１１に示すフィルタ係数を代入した配列filter[l][m]を乗算し累積するものである。

【００８９】図１３は、ハードウェアによるＬａｐｌａｃｉａｎフィルタの回路構成の一例である。まず、元の画像データの演算する単位領域から、データストリーム

として、例えば、ｐ[l][m]，ｐ[l-1][m]，ｐ[l][m-1]，ｐ[l][m+1]，ｐ[l+1][m]を構成し、時系列的にハードウェアモジュールの入力端ＩＮへ入力する。

【００９０】４クロックによって、前記のデータストリームを入力した時点で、画像データｐ[l][m]，ｐ[l-1][m]，ｐ[l][m-1]，ｐ[l][m+1]，ｐ[l+1][m]が、それぞれレジスタ４４、４３、４２、４１の出力端３４，３３，３２，３１および入力端ＩＮに現れている。そして、このとき、加算器４５，４６および４７による演算によって、加算器４７の出力端３８には、

ｐ[l-1][m]＋ｐ[l][m-1]＋ｐ[l][m+1]＋ｐ[l+1][m]

が出力されている。

【００９１】一方、シフトレジスタ５０により、その入力データが２ビット左シフトされることで、このシフトレジスタ５０の出力端３７には、

４×ｐ[l][m]

が出力される。

【００９２】そして、減算器４８により、シフトレジスタ５０の出力端３７のデータと、加算器４７の出力端３８のデータとの減算が行われ、この減算器４８の出力端３９には、

４×ｐ[l][m]－（ｐ[l-1][m]＋ｐ[l][m-1]＋ｐ[l][m+1]＋ｐ［ｌ＋１］［m］

が出力され、さらに加算器４９により、この出力端３９の出力データに、レジスタ４４の出力端３４の出力データｐ［ｌ］［m］が加算されることで、エッジ強調出力がＯＵＴに得られる。

【００９３】次に、図１４および図１５は、画像の単一画素の階調変換を行うフィルタを実現するソフトウェアで記述したルーチンＳｍＰＱＲ００１のプログラム例と、ハードウェアモジュールＨｍＰＱＲ００１の回路例を示す。このフィルタ処理を施すことにより、画像の濃度分布を変換してコントラストなどを補正することができる。

【００９４】図１４は、Ｃ言語で記述したルーチンの一例であり、次の式（２）で示す階調変換を用いている。

【００９５】

$$x(l,m)=a\times p(l,m)+b \qquad \text{式（２）}$$

図１５は、階調変換を実現する回路の一例である。この例では、ルックアップテーブル（ＬＵＴ）方式を用いている。すなわち、入力ＩＮに対し、テーブル６０を参照して、前記式（２）に対応する値ＯＵＴを出力する。

【００９６】テーブル６０は、ＳＲＡＭやＲＯＭなどのメモリ回路でテーブルデータをあらかじめ設定することで実現することができる。また、式（２）に従った入出力の真理値表を作って、アンドゲート、オアゲート、イクスクルーシブオアゲートなどの基本的ロジックゲートで、回路を構成することでテーブルを実現することもできる。

【００９７】この線形変換の動作は、元の画像データから、画素データｐ[l][m]を、順次、入力データＩＮとしてハードウェアモジュールに転送し、この入力データに対する出力データＯＵＴをテーブル６０を参照して求め、出力するものである。

【００９８】次に、図１６に、この実施例で示した２つの画像フィルタを連続して処理することにより、画像のコントラストを上げ色調補正をするという画像処理アプリケーションを実現する例のフローチャートを示す。

【００９９】まず、最初に、スキャナを使って、画像の読み取りに必要な解像度と階調数で紙文書を読み取り、画像データとして蓄積する（ステップＳ４０１）。画像データは、例えば縦×横＝Ｌ×Ｍ画素のＲＧＢ画像とす

る。この場合、画像データは、Ｒプレーン、Ｇプレーン、Ｂプレーンのそれぞれにおいて、処理が行われる。

【０１００】蓄積された画像データは、前述の実施例で述べたＬａｐｌａｃｉａｎフィルタにより、同じハードウェアモジュールでＲ１，Ｇ１，Ｂ１，Ｒ２，Ｇ２，Ｂ２…の順でエッジ強調処理を行う（ステップＳ４０２）。

【０１０１】つぎに、前述の実施例で述べた階調変換フィルタによって、Ｌ×Ｍ画素のそれぞれについて、Ｒプレーン、Ｇプレーン、Ｂプレーンのそれぞれにおいて、階調補正を行う（ステップＳ４０３～Ｓ４１１）。

【０１０２】この場合、適正な色バランスを持った画像に仕上げるために、通常、Ｒ，Ｇ，Ｂの各色ごとに異なる階調変換カーブを使うので、異なるハードウェアモジュールにより、やはり、Ｒ１，Ｇ１，Ｂ１，Ｒ２，Ｇ２，Ｂ２…の順で各画素の階調変換処理を行う。

【０１０３】階調変換処理は、回路規模的に比較的小さいものなので、各色の階調変換処理ハードウェアモジュールは、３つが同時にプログラマブル論理回路に搭載可能である。したがって、前述のようにして、予めハードウェアモジュールが結合された後、各画素の階調変換が行われるものである。

【０１０４】以上のように、上述の実施例では、結合モジュールにより処理が行われるため、Ｒ，Ｇ，Ｂ階調変換用のハードウエアモジュールのコンフィギュレーションは、一回でよく、全ての画素の処理の処理を高速に終了して、コントラストが高く色バランスのよい画像が、高速に得られる。

【０１０５】なお、上述の実施の形態では、プログラムのヘッダ部にハードウエアモジュールの識別符号と、それぞれのサイズの情報を記述するようにしたが、アプリケーションプログラムを解析することで、それらを検出するようにしてもよい。

【０１０６】また、前処理ルーチンとしての組み合わせ検出および結合処理は、アプリケーションプログラム上に記述しておいてもよい。

【０１０７】

【発明の効果】以上説明したように、この発明によれば、少なくとも処理の一部分が回路構成を再構成できるプログラマブル論理回路で処理される情報処理システムにおいて、アプリケーションプログラム中で順序が連続するハードウェアモジュール処理のうちプログラマブル論理回路に同時に回路を再構成できるものを、ひとつのコンフィギュレーションデータに結合することにより、アプリケーション全体の処理時間のうち回路を再構成する時間を最小にすることができ、情報処理システムの最高の処理能力を発揮することが可能となる。

【図面の簡単な説明】

【図１】この発明による情報処理システムの実施の形態の概念構成を示すブロック図である。

【図２】この発明による情報処理システムの実施の形態で用いるアプリケーションプログラムの一例を説明するための図である。

【図３】この発明による情報処理システムの実施の形態のハードウエア構成例を示す図である。

【図４】プログラマブル論理回路の一例を説明するための図である。

【図５】プログラマブル論理回路の一例を説明するための図である。

【図６】この発明による情報処理システムの実施の形態の主要な処理動作を説明するためのフローチャートである。

【図７】この発明による情報処理システムの要部の動作を説明するために用いる図である。

【図８】この発明による情報処理システムの要部の動作を説明するために用いる図である。

【図９】図７の一部の処理ルーチンを説明するためのフローチャートである。

【図１０】この発明による情報処理システムの処理の具体例の実行シーケンスを示す図である。

【図１１】この発明による情報処理システムの処理の具体例で用いるハードウエアモジュールの一つを説明するための図である。

【図１２】図１１のハードウエアモジュールをソフトウエアで記述したときの例を示す図である。

【図１３】図１１のハードウエアモジュールでプログラマブル論理回路に構成される回路例を示す図である。

【図１４】この発明の実施の形態で用いるハードウエアモジュールの他の一つを説明するための図である。

【図１５】図１５のハードウエアモジュールでプログラマブル論理回路に構成される回路例を示す図である。

【図１６】この発明の実施の形態が適用される画像処理の流れを示すフローチャートである。

【図１７】従来の情報処理システムの一つを説明するための図である。

【符号の説明】

１０　情報処理システム
１１　ＣＰＵ
１２　チップセット
１３　メインメモリ
１４　バス
１５　プログラマブル論理回路インターフェース
１６　プログラマブル論理回路
１７　ハードディスクインターフェース
１８　ハードディスクドライブ
１９　通信インターフェース
２０　ネットワーク
２１、２２、２３　ネットワーク上の記憶装置
１００　アプリケーションプログラム
１６０　コンフィギュレーションメモリ

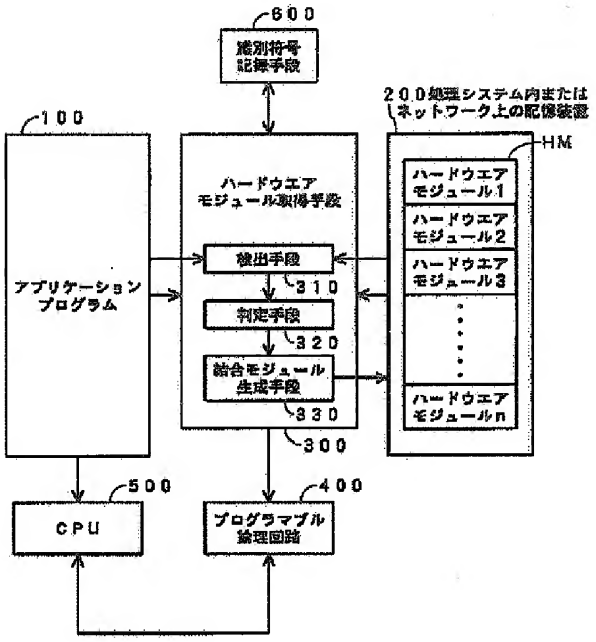
17

２００　ネットワーク上の記憶装置
３００　ハードウエアモジュール取得手段
３１０　検出手段
３２０　判定手段

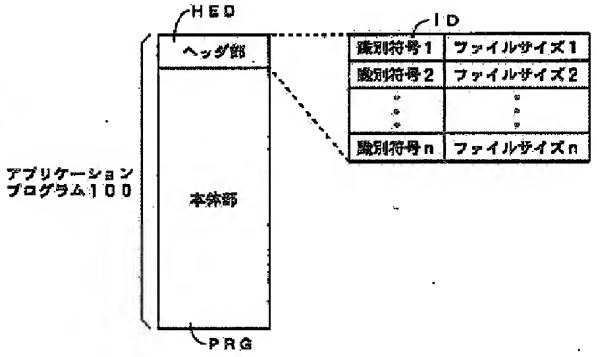
18

３３０　結合モジュール生成手段
４００　プログラマブル論理回路
５００　ＣＰＵ
６００　識別符号記憶手段

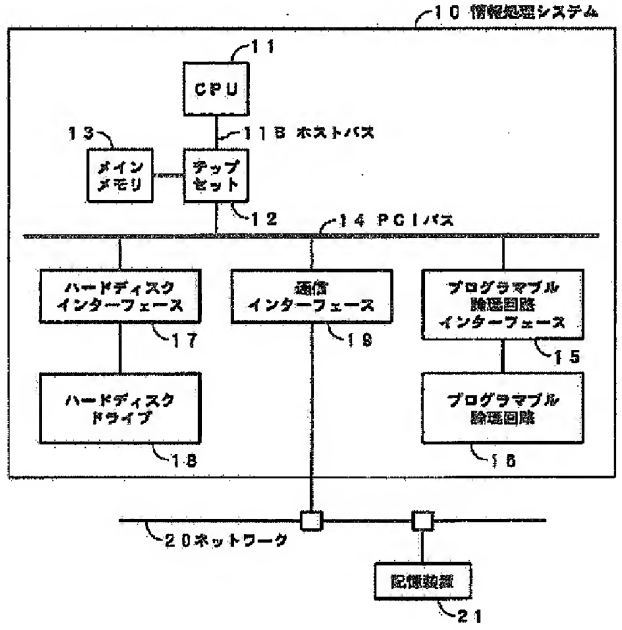
【図１】



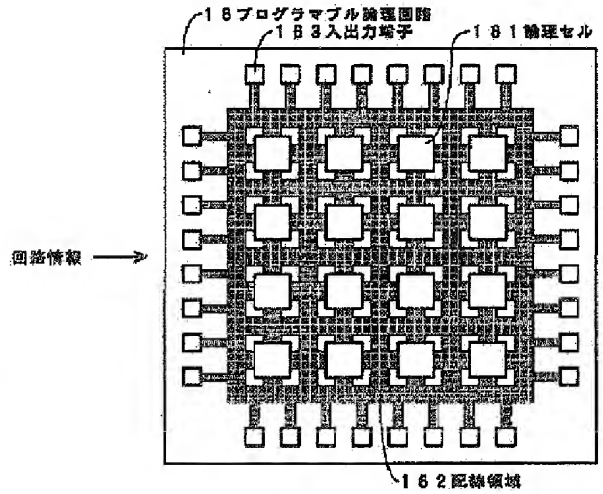
【図２】



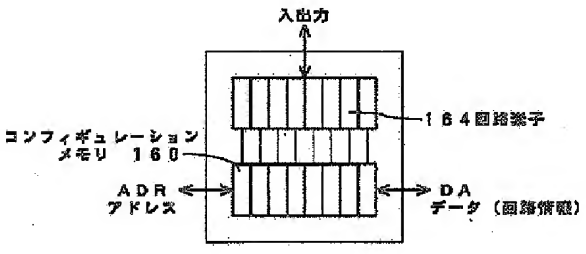
【図３】



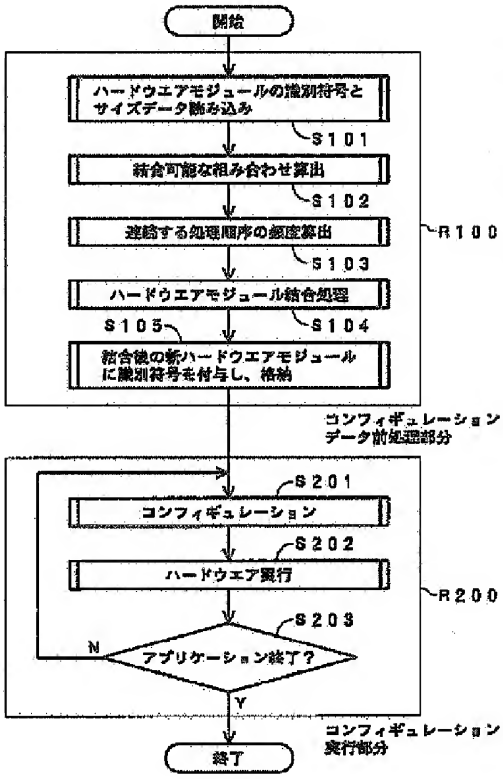
【図４】



【図５】



【図６】

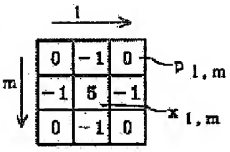


【図７】

TBL1 結合可能テーブル

連続数	No.	組み合わせ名	結合後サイズ＜最大搭載規模？
2	1	AB	○
	2	AC	○
	3	AD	×
	4	AE	×
・・・	・・・	・・・	・・・
3	1	ABC	○
	2	ABD	×
	3	ABE	×
・・・	・・・	・・・	・・・

【図１１】



【図１２】

```
Int  SmXYZ001(int p)
{
int filter[3][3]={0,-1,0,-1,5,-1,0,-1,0};
    x=0;
    for(l=-1;l<=+1;l++){
       for(m=-1;m<=+1;m++){
       x+=p[l][m]*filter[l][m];
       }
    }

    return x;
}
```

【図１４】

```
Int  SmPQR001(int p)
{
     y=f(p[l][m])
     return y;
}
```

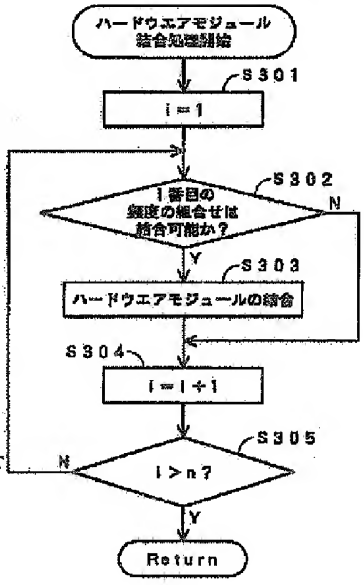
【図８】

TBL2 頻度テーブル

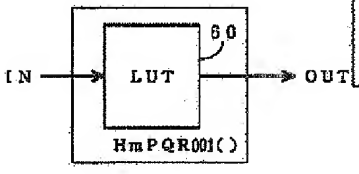
No.	組み合わせ名	頻度
1	AB	100
2	AC	40
3	ABD	10
4	ABC	5
・・・	・・・	・・・
n		

【図９】

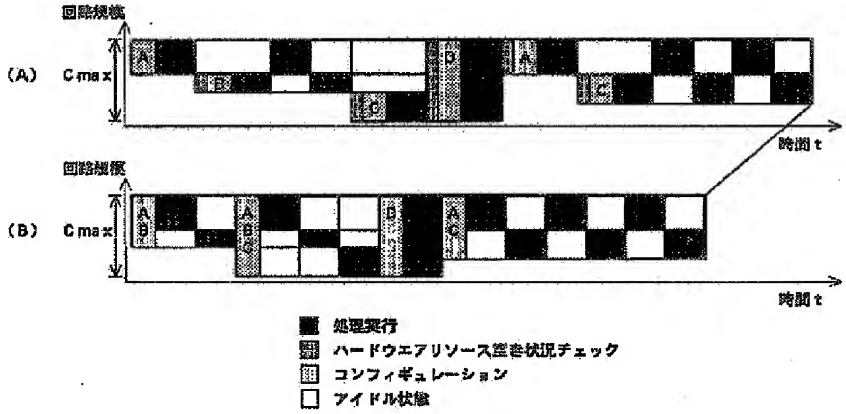
ハードウエアモジュール結合処理のルーチン



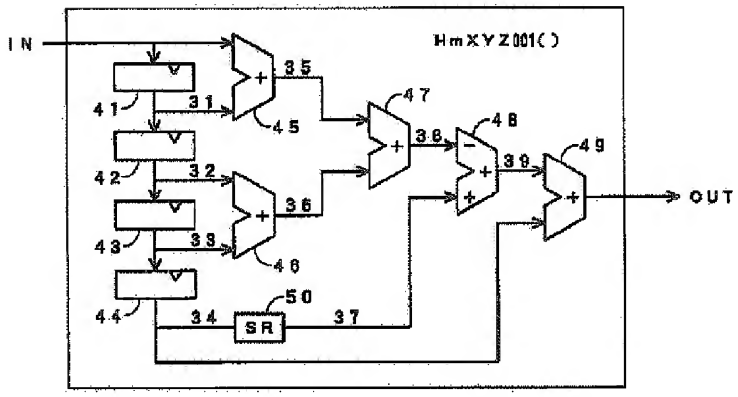
【図１５】



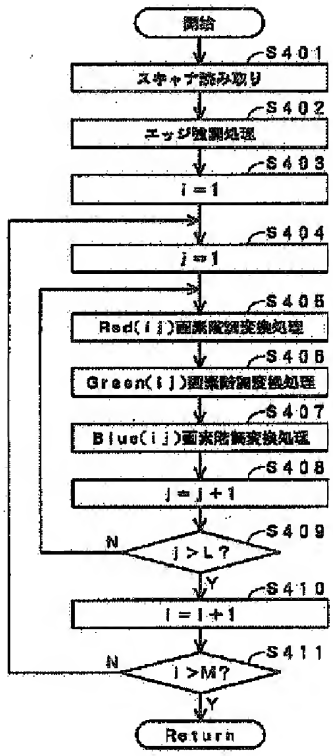
【図１０】



【図１３】



【図１６】



【図１７】